Record-Level Access: Under the Hood

Salesforce, Spring '22

@salesforcedocs

Last updated: February 24, 2022

CONTENTS

RECORD-LEVEL ACCESS: UNDER THE HOOD

Preface

If you're a Salesforce architect, you already know your overarching goal for data security: Understand your company's enterprise security model, then build a data access model that reflects it in your Salesforce organization. You have a rich set of Salesforce tools at your disposal for managing that data access, but you don't manage it at just one level. Instead, you must carefully consider when and how to control access at the object, record, and field levels.

In this paper, you can find a brief overview of these different levels of data access, and under the hood, table-level views of record-level access. After reading *Record-Level Access: Under the Hood*, you should feel better prepared to give the right users the right access to the right records at the right time — and do so in the fastest time possible.

Audience

This paper is intended for expert architects working on Salesforce implementations that have complex record access requirements or require large-scale realignments of sales organizations.

Assumptions

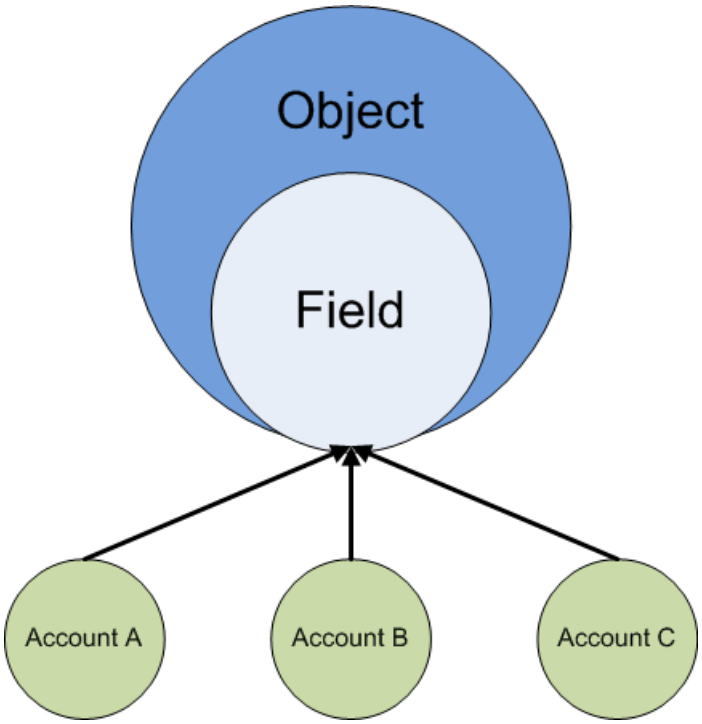
This paper assumes that you are an expert in Salesforce administration and security, and in SQL and relational database concepts.

Data Access in Salesforce

To meet your company's security needs, it's important to understand what data access means to your users and to you.

Data Access: User's Perspective

If you put yourself in your users' shoes, you won't necessarily know or care how you're getting access to records, but you might want to understand what having access means within the context of your organization. The following graph can help users visualize the different kinds of access that can be configured in Salesforce.



For example, if a user has access to an account field, then they have access to both the account field and the account object itself. However, a specific account record, such as "Account A", might not be accessible to that user due to additional access control applied via sharing rules or other tools.

Data Access: Architect's Perspective

As an architect, you must both understand your user's perspective and know how to grant users only the appropriate level of access to the data that they should be able to access.

From an architect's perspective, data access in Salesforce falls into two main categories: object-level access, which includes field-level access, and record-level access.

Object-level access determines whether a user has access to a particular object, which fields they can see on that object, and which actions they can perform. You configure object level access on user profiles.

Restricting access

The "Read," "Create," "Edit," and "Delete" object permissions determine which actions a user can perform on any of the object's records to which they have access. Field-Level Security allows you to prevent certain users from seeing sensitive or confidential information contained in records they can see.

Opening up access

The "View All" and "Modify All" object permissions give users access to all of an object's records, regardless of record-level access settings.

Record-level access (called "Sharing" in Salesforce) determines which records a user can see for a particular object, using the following tools:

- Organization-wide defaults
- Role hierarchy

- Territory hierarchy
- Sharing rules
- Teams
- Manual sharing
- Programmatic sharing

Because you have so many options for managing record-level access — and because some of these options are affected by organizational dependencies — determining which records users can access can quickly become complicated. Additionally, you might also be changing your sharing configuration frequently in response to new business requirements. This can trigger record access changes that ripple through your organization. These changes have an even greater impact in very large organizations, where it can take some time to recalculate access for a large number of users, and adjust the tables that record their access rights. For these reasons, it's important to understand how Salesforce calculates and grants access at the database level.

Record Access Calculation

Every time a user attempts to open a record, run a report, access a list view, or search for data using the user interface or API, Salesforce checks the configuration of its record access features to determine which records the user can access. These configurations can be elaborate, especially in large organizations with hundreds of hierarchy nodes, thousands of sharing rules, millions of data rows, and portals for customers and business partners. Processing such dissimilar data and complex relationships would require far more time than the 300-millisecond Salesforce benchmark for rendering pages.

Rather than applying every sharing rule, traversing all hierarchies, and analyzing record access inheritance in real time, Salesforce calculates record access data only when configuration changes occur. The calculated results persist in a way that facilitates rapid scanning and minimizes the number of database table joins necessary to determine record access at run time.

Access Grants

When an object has its organization-wide default set to Private or Public Read Only, Salesforce uses access grants to define how much access a user or group has to that object's records. Each access grant gives a specific user or group access to a specific record. It also records the type of sharing tool — sharing rule, team, etc. — used to provide that access. Salesforce uses four types of access grants: explicit grants, group membership grants, inherited grants, and implicit grants.

Explicit Grants

Salesforce uses explicit grants when records are shared directly to users or groups. Specifically, Salesforce uses explicit grants when:

- A user or a queue becomes the owner of a record.
- A sharing rule shares the record to a personal or public group, a queue, a role, or a territory.
- An assignment rule shares the record to a user or a queue.
- A territory assignment rule shares the record to a territory.
- A user manually shares the record to a user, a personal or public group, a queue, a role, or a territory.
- A user becomes part of a team for an account, opportunity, or case.
- A programmatic customization shares the record to a user, a personal or public group, a queue, a role, or a territory.

Note: If your organization doesn't have an efficient sharing architecture, it might encounter performance problems when you use automated processes that generate a very large number of explicit grants, such as major sales realignments.

Group Membership Grants

Grants that occur when a user, personal or public group, queue, role, or territory is a member of a group that has explicit access to the record. For example, if a sharing rule explicitly grants the Strategy group access to the Acme record, and Bob is a member of the Strategy group, Bob's membership in the Strategy group grants him access to the Acme record.

Inherited Grants

Grants that occur when a user, personal or public group, queue, role, or territory inherits access through a role or territory hierarchy, or is a member of a group that inherits access through a group hierarchy.

Implicit Grants

Grants that occur when non-configurable record-sharing behaviors built into Salesforce Sales, Service, and Portal applications grant access to certain parent and child records. For example, with this default logic, sometimes referred to as *built-in sharing*, users can view a parent account record if they have access to its child opportunity, case, or contact record. If those users have access to a parent account record, they can also access its child opportunity, case, and contact records.

Database Architecture

Important: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Salesforce stores access grants in three types of tables.

Object Record Tables

Tables that store the records of a specific object, and indicate which user, group, or queue owns each record.

Object Sharing Tables

Tables that store the data that supports explicit and implicit grants. Most objects in your organization (to see them, from Setup, enter `Sharing Settings` in the Quick Find box, then select **Sharing Settings**) get their own Object Sharing table, unless any of the following conditions are also true:

- The object is a detail in a master-detail relationship. In master-detail relationships, the Object Sharing table for the master object controls access to the detail object.
- Both organization-wide default settings (internal and external) are Public Read/Write.
- The object is of a type that doesn't support Object Sharing tables, such as Activities or Files. These objects have their own access control mechanism.

Group Maintenance Tables

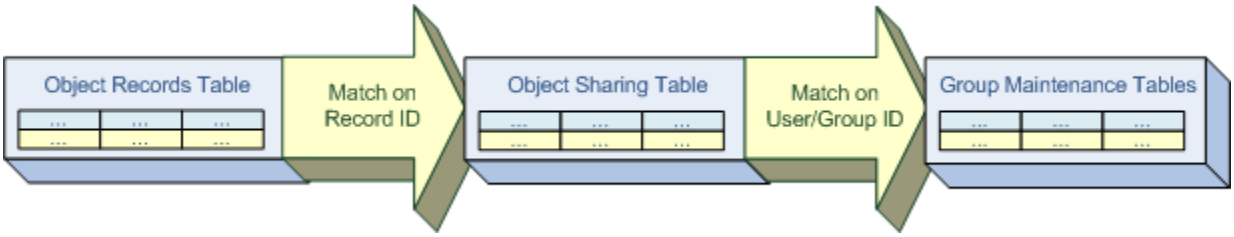
Tables that store the data supporting group membership and inherited access grants. For example, if the Object Sharing table grants Bob explicit access to the Acme account record, Salesforce checks the Group Maintenance tables to see which users inherit record access from Bob and grants users access to the Acme record. These grants are established in advance when you create or change the group (or role, or territory) membership information.

While Object Sharing tables store access grants to individuals and groups, Group Maintenance tables store the list of users or groups that belong to each group, indicating *group membership*. Both types of tables are used to determine a user's access to data when they are searching, querying, or pulling up a report or list view.

When a user tries to retrieve one or more records, Salesforce generates a SQL statement that searches the Object Record table for records matching the user's search string. If the record exists, Salesforce appends SQL to the statement that joins the Object Records table with the Object Sharing table, and the Object Sharing table with the Group Maintenance tables. Salesforce queries the joined tables for access grants that give the querying user access to the record.

When you join the following tables...	Salesforce matches on
Object Record and Object Sharing	Record ID
Object Sharing and Group Maintenance	User ID or group ID

This diagram illustrates that Salesforce matching process in sequence.



Salesforce returns only records that satisfy the entire statement, including its appended SQL. To satisfy the statement, the record must exist, and either the Object Sharing table or the Group Maintenance tables must grant access to the querying user.

Even though both the Object Sharing and Group Maintenance tables provide access grants, the ways in which they provide those grants differ significantly.

- Object Sharing tables simply store each access grant in separate rows called *sharing rows*, each of which grants a user or group access to a particular record.
- Group Maintenance tables are more complex because a single group membership or inherited access grant can give several users and groups multiple ways to access a record.

Sharing Rows

Each sharing row includes the:

- ID of the record to which the row grants access
- ID of the user or group to whom the row grants access
- Level of access the row allows, such as Read Only or Full Access
- *Row cause*, which indicates the reason Salesforce grants the user or group access to the record

  For example, when a record owner manually shares a record with a user or group, Salesforce creates a sharing row with a `Manual` row cause. When a sharing rule shares the record with a user or group, Salesforce creates a sharing row with a `Rule` row cause.

The simplified tables in the following examples demonstrate how Salesforce creates sharing rows under the hood.

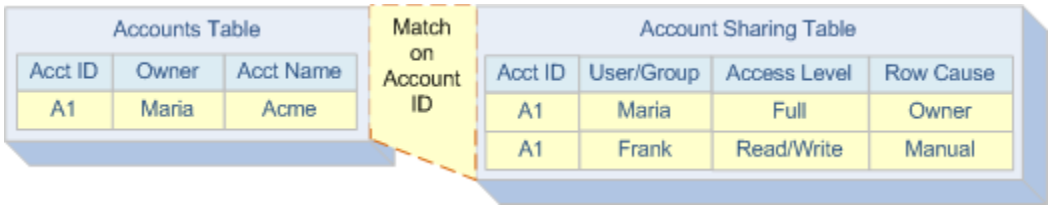
Note: For readability, these tables do not contain all of the actual database values and structure.

Example 1

Maria, whose role is Sales Executive, creates an Account record (ID=A1) for a company called "Acme." Under the hood, Salesforce creates a sharing row for her as the record owner in the Account Sharing table, the Object Sharing table for the Account object.

Example 2

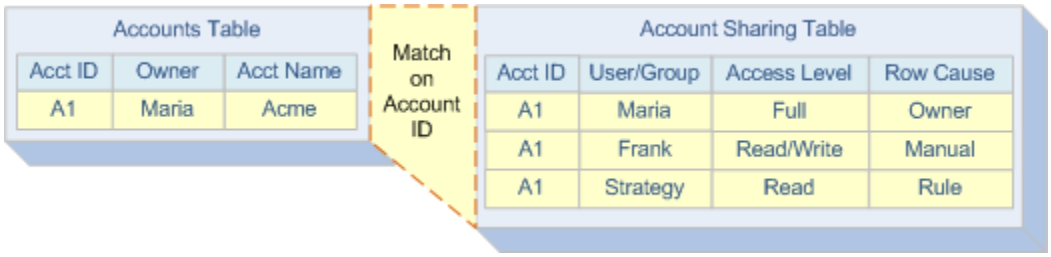
Maria manually shares the Acme account record with Frank, the services executive. Under the hood, Salesforce adds a sharing row for Frank.



While only one account record exists for Acme, the Account Sharing table now contains two entries for the Acme record. This update happens because Salesforce grants access to the Acme account record twice: once to Maria as the owner and once to Frank.

Example 3

An administrator creates a sharing rule that shares the Sales Executive's records with the Strategy group, giving them Read Only access. Under the hood, Salesforce adds a sharing row that gives the Strategy group access to Maria's Acme account record.



For users with multiple access grants to a record, Salesforce uses the most permissive grant when determining record access. For example, if Frank joins the Strategy group, he still maintains the Read/Write access that Maria granted him in Example 2.

If many users have the same record access requirements, it's efficient to place those users in a group and grant access to the group instead of to the individuals. This practice saves time and results in fewer sharing rows, thus reducing your organization's record access data volume.

Group Maintenance Tables

Sharing rows grant access to users and groups, but the data that specifies who belongs to each group resides in the Group Maintenance tables. These tables store membership data for every Salesforce group, including *system-defined groups*. System-defined groups are groups of users that Salesforce creates and manages internally to support various features and behaviors, such as queues. This type of management lets the data that supports queues and personal or public groups coexist in the same database tables, and unifies how Salesforce manages the data. For example, Salesforce can grant record access to a queue the same way it grants record access to a public group.

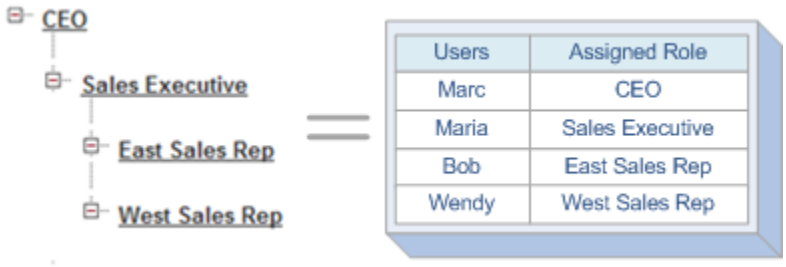
Salesforce also uses system-defined groups to implement hierarchies. During recalculation, Salesforce creates two types of system-defined groups, *Role* groups and *RoleAndSubordinates* groups, for every node in the role hierarchy. If the organization has external organization-wide defaults enabled, a third type of system-defined group, *RoleAndInternalSubordinates*, is created.

Group	Consists of	Purpose
Role	Users assigned to any of the following. • A specific role • One of its manager roles	Used to give managers access to their subordinates' records
RoleAndSubordinates	Users assigned to any of the following. • A specific role • One of its manager roles • One of its subordinate roles	Used when an organization defines a rule that shares a set of records with: • A particular role • Its subordinates
RoleAndInternalSubordinates	Users assigned to any of the following. • A specific role • One of its manager roles • One of its subordinate roles, excluding Portal roles	Used when an organization defines a rule that shares a set of records with: • A particular role • Its subordinates, excluding Portal roles

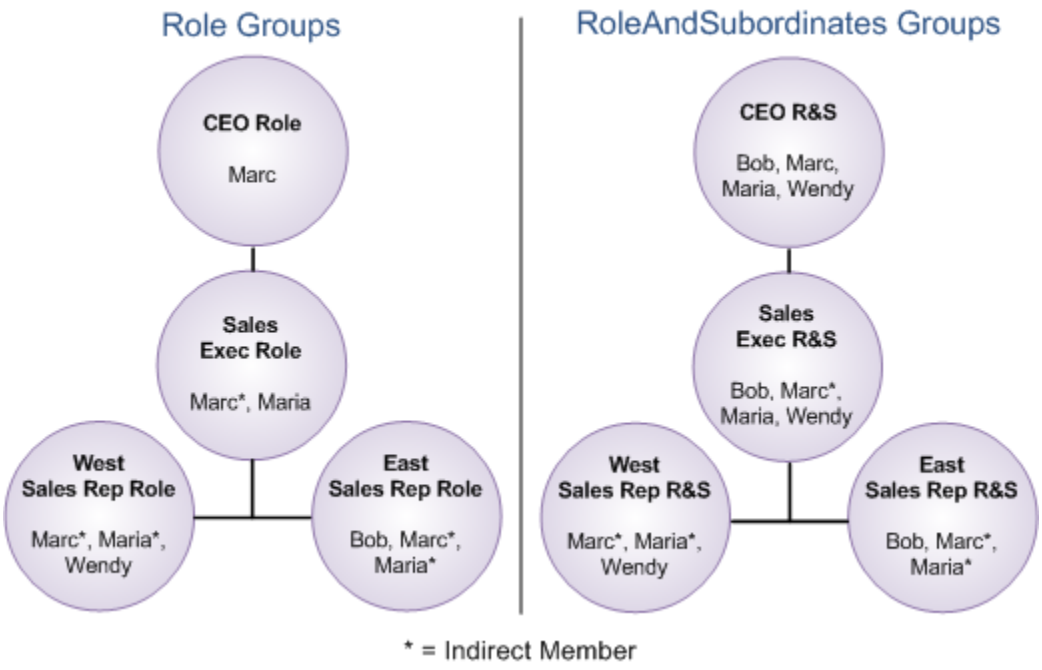
All three group types have:

- Indirect members, who inherit record access from the group's direct members and are assigned to manager roles
- Direct members, who are defined according to their group type
  - In Role groups, direct members are those members assigned to the role the group represents.
  - In RoleAndSubordinates groups, direct members are those members assigned to the role the group represents or one of its subordinate roles
  - In RoleAndInternalSubordinates groups, direct members are those members assigned to the role the group represents, or one of its non-portal subordinate roles.

For example, consider the following simple role hierarchy.



To support the record access inheritance this hierarchy establishes, Salesforce defines the following Role and RoleAndSubordinates groups, resulting in eight total groups.



By scanning the Role groups, Salesforce can quickly identify the indirect members who inherit record access from users at that role. For example, to see which users inherit record access from Bob in the role hierarchy, Salesforce simply searches for Role groups that have Bob as a direct member (the East Sales Rep Role group), and finds all the indirect members in those groups (Marc and Maria).

Likewise, by scanning the RoleAndSubordinates groups, Salesforce can quickly see which users receive access through role and subordinate sharing rules. For example, if a rule shares a set of records with users in the Sales Executive role and their subordinates, Salesforce can identify those users by scanning the Sales Executive RoleAndSubordinates group.

System-defined groups support Territory Management in a similar way.

For each territory, Salesforce creates a:

- Territory group, in which users who are assigned to the territory are direct members, while users assigned to territories higher in the hierarchy are indirect members
- TerritoryAndSubordinates group, in which users who are assigned to that territory or territories lower in the hierarchy are direct members, while users assigned to territories higher in that branch are indirect members

Without system-defined groups, every record access attempt would have to send Salesforce traversing every branch in its hierarchies, jumping back and forth between tables of user data and tables of hierarchical data. Salesforce would also have to use disparate processes to handle what are all essentially collections of users.

Users can't modify system-defined groups through the user interface or API in the ways that they can personal and public groups; however, modifying a queue or hierarchy causes Salesforce to recalculate its associated system-defined groups accordingly. Thus, the size and complexity of an organization's queues and hierarchies directly affect the duration of record access calculations.

Sample Scenarios

The following scenarios use simplified tables to illustrate how Salesforce recalculates the Object Sharing and Group Maintenance tables according to various record access changes, and uses those calculations to determine record access. Yellow highlights indicate data that grants access to the sample account record.

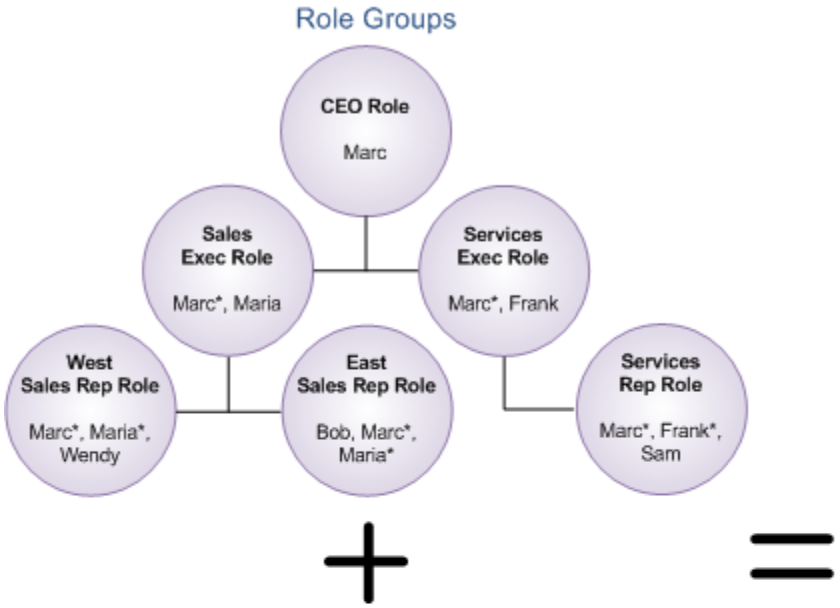
Note: The scenarios only show data that's essential to the example; they don't show all of the fields and tables Salesforce uses for calculating record access.

In these scenarios, the organization-wide default settings are Private for all objects, and the role hierarchy and users are as follows.

Users

Users	Assigned Role
Marc	CEO
Maria	Sales Executive
Frank	Services Executive
Bob	East Sales Rep
Wendy	West Sales Rep
Sam	Services Rep

Salesforce generates the following groups to support the record access inheritance the role hierarchy establishes.



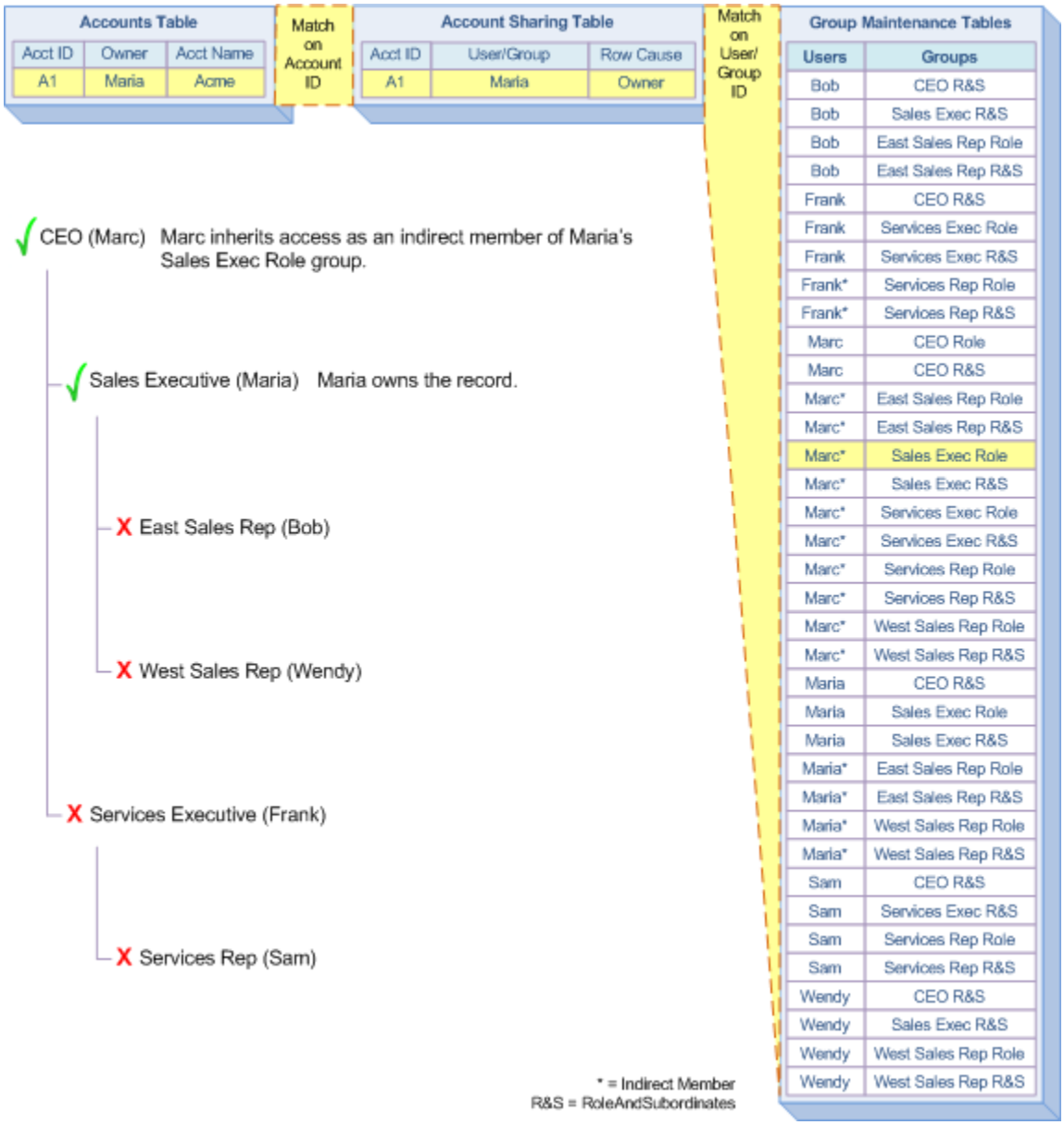
Group Maintenance Tables

Users	Groups
Bob	CEO R&S
Bob	Sales Exec R&S
Bob	East Sales Rep Role
Bob	East Sales Rep R&S
Frank	CEO R&S
Frank	Services Exec Role
Frank	Services Exec R&S
Frank*	Services Rep Role
Frank*	Services Rep R&S
Marc	CEO Role
Marc	CEO R&S
Marc*	East Sales Rep Role
Marc*	East Sales Rep R&S
Marc*	Sales Exec Role
Marc*	Sales Exec R&S
Marc*	Services Exec Role
Marc*	Services Exec R&S
Marc*	Services Rep Role
Marc*	Services Rep R&S
Marc*	West Sales Rep Role
Marc*	West Sales Rep R&S
Maria	CEO R&S
Maria	Sales Exec Role
Maria	Sales Exec R&S
Maria*	East Sales Rep Role
Maria*	East Sales Rep R&S
Maria*	West Sales Rep Role
Maria*	West Sales Rep R&S
Sam	CEO R&S
Sam	Services Exec R&S
Sam	Services Rep Role
Sam	Services Rep R&S
Wendy	CEO R&S
Wendy	Sales Exec R&S
Wendy	West Sales Rep Role
Wendy	West Sales Rep R&S

* = Indirect Member
R&S = RoleAndSubordinates

Scenario 1

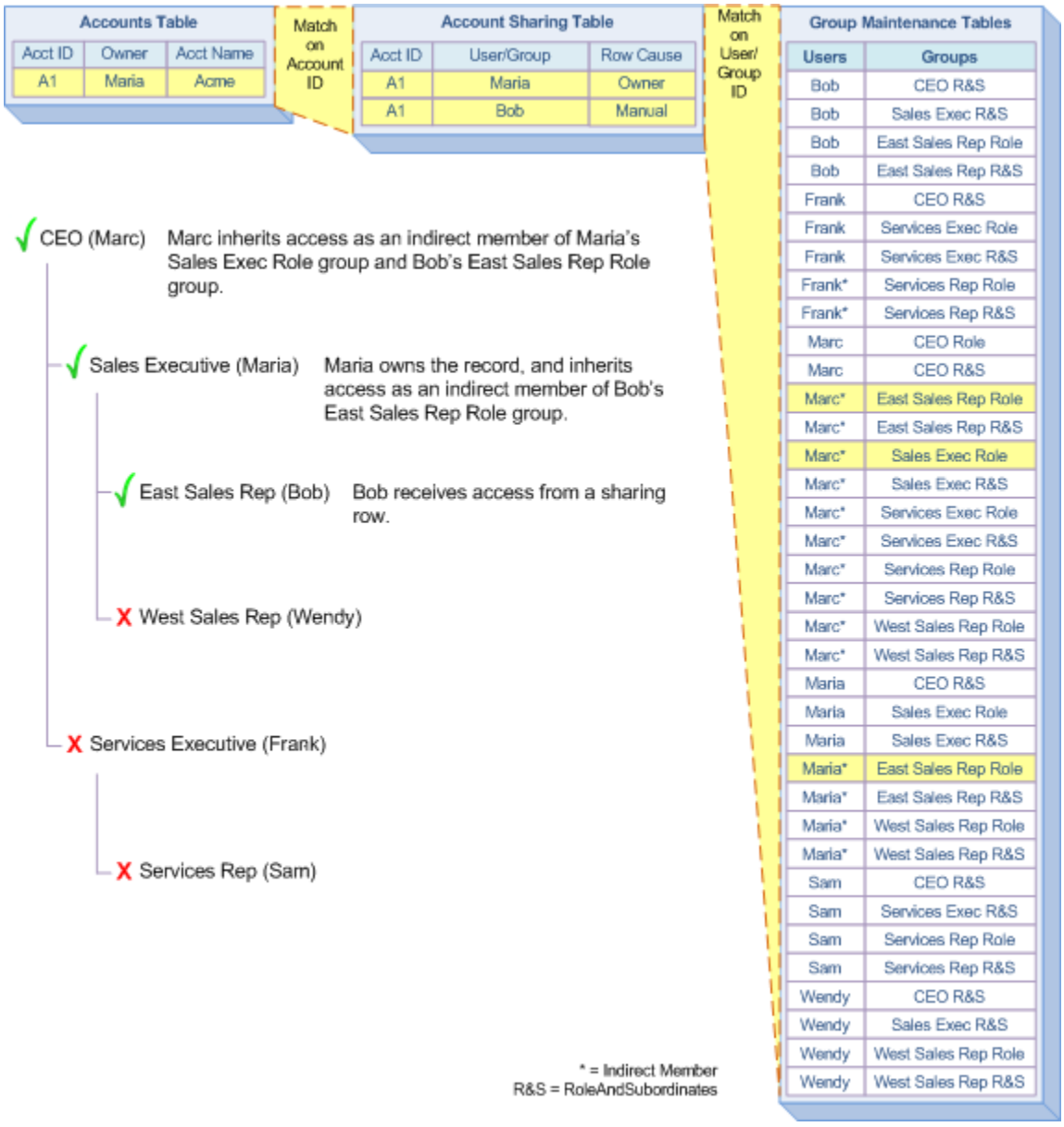
Maria creates an account record (A1) for Acme. Under the hood, Salesforce creates a new account record for Acme and an owner sharing row for Maria in the Account Sharing table. Marc, because he is above Maria in the role hierarchy and an indirect member of Maria's Sales Executive role group, inherits access.



Scenario 2

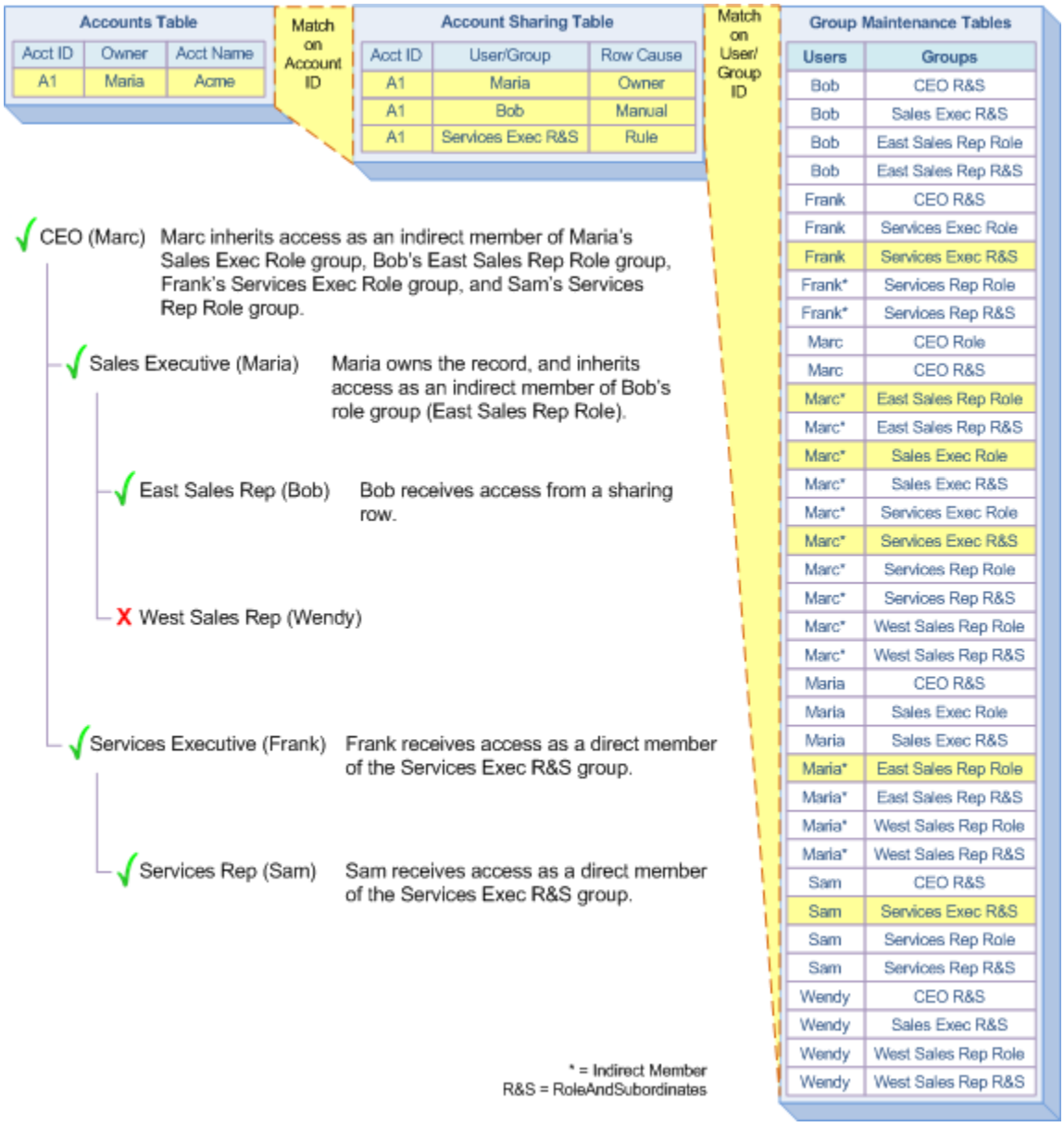
Maria manually shares the Acme account record with Bob. Under the hood, Salesforce creates a sharing row in the Account Sharing table for Bob. Both Maria and Marc have access to the record, because they are above Bob in the role hierarchy and indirect members of Bob's East Sales Rep role group.

For manual record sharing, programmatic record sharing, and team sharing, the Object Sharing table creates rows the same way but with different row causes.



Scenario 3

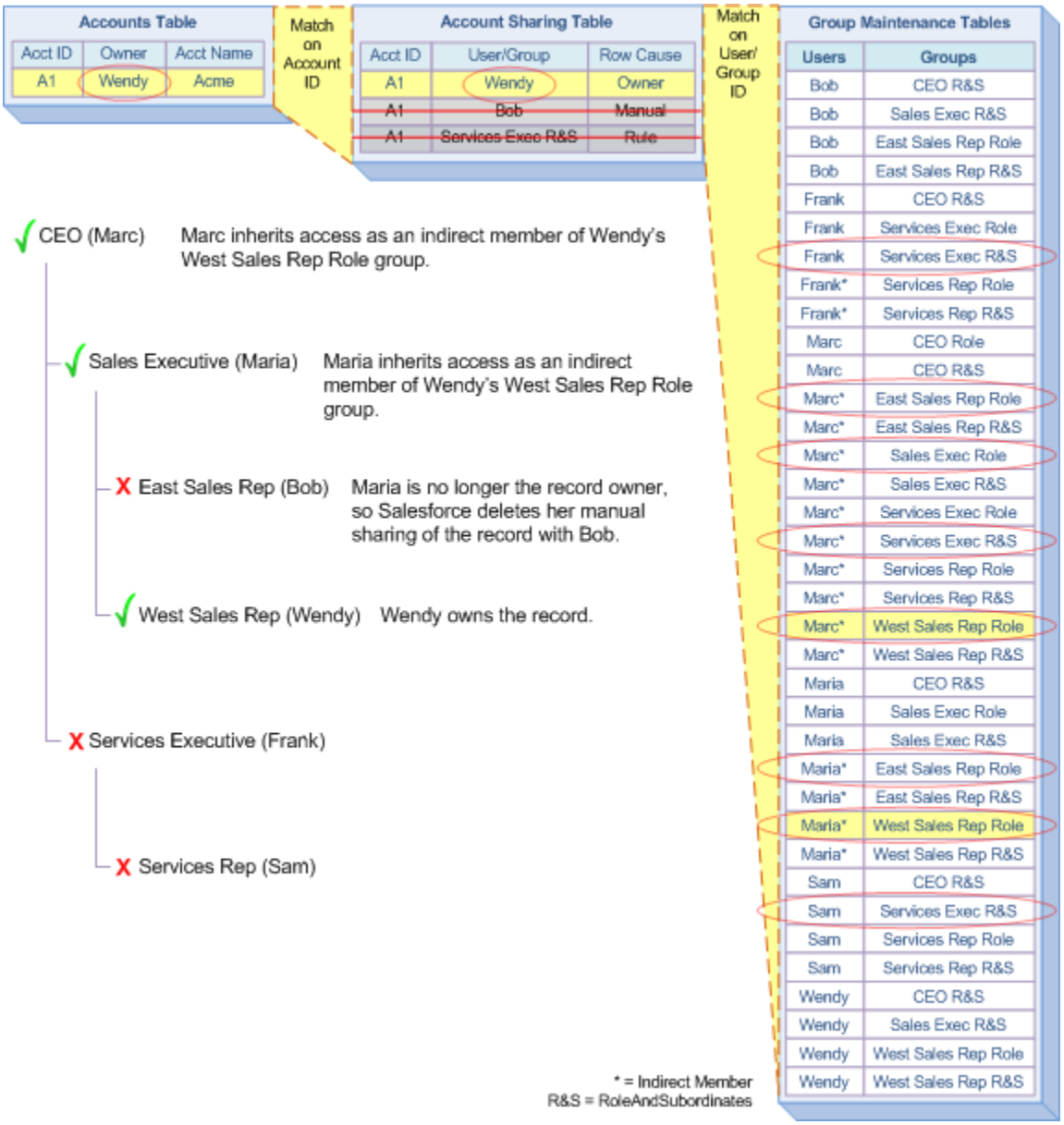
An administrator creates a sharing rule that shares the Sales Executive's records with the users in the Services Executive role and their subordinates. Under the hood, Salesforce creates a sharing row in the Account Sharing table for the Services Exec RoleAndSubordinates group, giving Frank and Sam access to the Acme record.



Scenario 4

Maria changes the owner of the Acme record to Wendy. When a record owner changes, Salesforce deletes its associated sharing rows with Manual row causes, so Bob loses access to the record. Also, because Maria, the Sales Executive, no longer owns the record, the rule from Scenario 3 no longer applies. Under the hood, Salesforce deletes the sharing row for the Services Exec RoleAndSubordinates group from Scenario 3, causing Frank and Sam to lose access to the Acme record. Salesforce also replaces Maria's name with Wendy's in the Account Sharing table.

The red ovals in this diagram indicate the many fields this seemingly minor change affects.



Putting It All Together

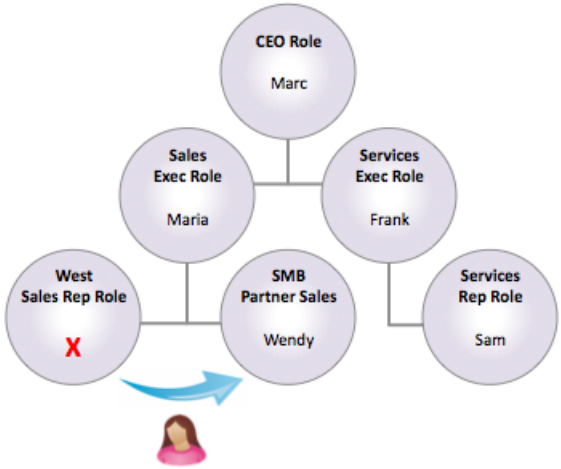
In this paper's scenarios, you have seen how the access rights of users are recorded in the Object Sharing and Group Maintenance tables of the Salesforce platform. In those relatively simple examples, only a few rows were changing in many of the tables. However, many common administrative operations might require a much more substantial recalculation of access and many more updates to the underlying tables. This level of complexity is especially true for organizations with large amounts of data and complex role hierarchies. To illustrate how involved those operations might get, we will consider one seemingly simple group operation that can have a major impact on sharing recalculations: changing a user's role. As in the previous scenarios, the organization-wide default settings are Private for all objects.

The role hierarchy and users are mostly the same, with two differences.

- A new role has been created for the Small and Medium Business Partner Sales organization.

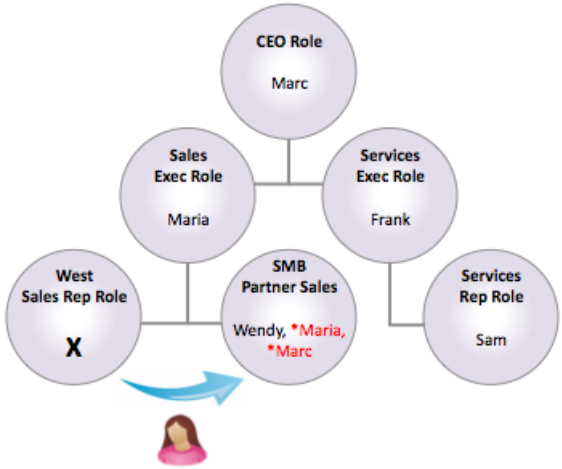
- Instead of a broad sharing rule providing access to all Sales data to the Services branch, there is a more focused sharing rule providing access only to data from the West Sales Rep Role—the SMB Partner data is not shared to Services.

In this scenario, Wendy moves from the West Sales Rep role to the new SMB Partner Sales role, which is located in a separate branch of the role hierarchy.

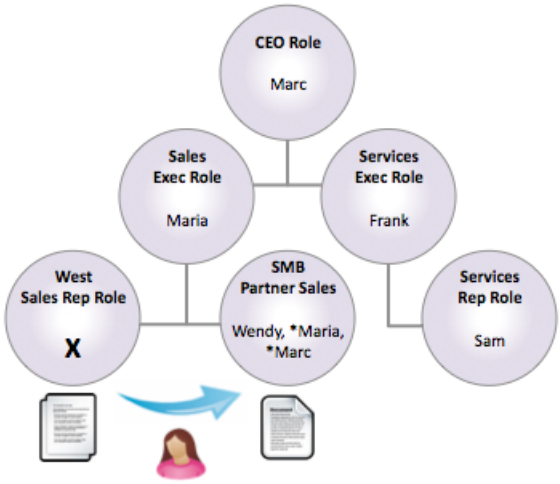


When Wendy makes her move, the underlying sharing system performs the following maintenance under the hood.

- If Wendy is the first member in her new role to own any data, Salesforce arranges access to her data for all users in roles above her in the hierarchy. This arrangement is performed by making those users indirect members of Wendy's new role. Note the inclusion of Maria and Marc in the new SMB Partner Sales role.



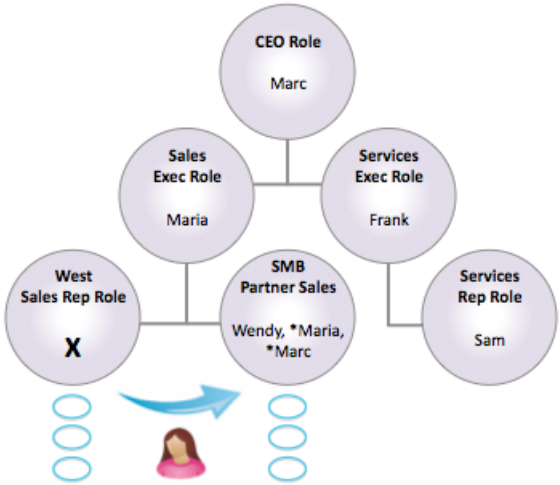
- If Wendy's new role has different settings than her old role for access to child records of Account—Contacts, Cases, and Opportunities—Salesforce:
  - Removes some of Wendy's shares to those records
  - Adds new shares to reflect the change in settings



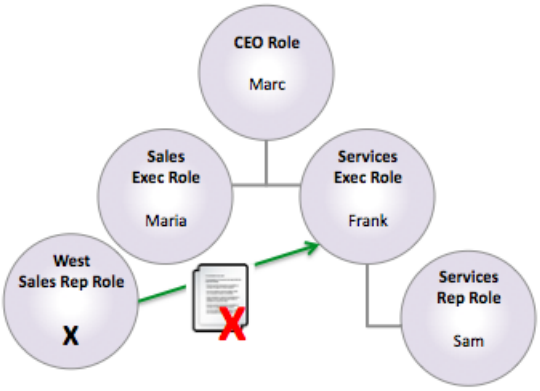
- If Wendy owns any accounts that have been enabled for either the Customer or Partner portals, Salesforce:
  - Makes changes to group membership
  - Adjusts shares that provide access in the hierarchy to records owned by or shared to portal users

  For each portal-enabled account, 1–3 roles are appended to the main hierarchy below the account owner's role. When Wendy moves, Salesforce removes these portal roles from her old role and appends them to her new role for *every* portal-enabled account she owns.

  Note: Salesforce must delete shares written to Wendy's old role—shares that provide her and all her superiors with access to records owned by or visible to members of each portal account—and add them to her new role.



- To reflect Wendy's new assignment, Salesforce recalculates all sharing rules that include Wendy's old or new role in the source group.



Specifically, Salesforce deletes shares to all of Wendy's records from:

- The top role in the Service branch of the hierarchy
- All of its subordinates

Depending on the sharing rule settings for her accounts, Salesforce might also need to remove shares to her account child records.

Note: If Wendy owns portal accounts, and there are sharing rules that use portal roles as the source group, Salesforce may need to recalculate these rules. Those rules may also no longer be valid given Wendy's new location in the hierarchy, in which case an administrator may need to modify or delete them.

In addition to these changes, the managers in the branch above Wendy's old role lose access to all the data that she owns, as well as to child records shared through the role settings. This process is part of the normal operation of inheritance in the hierarchy—no updates to group membership or sharing tables are required.

So a lot can go on under the hood when an administrator takes what looks like a simple action, such as changing a user's role. We chose a particularly complex operation to illustrate all the possible types of sharing maintenance, but other common group and data updates can have a similar impact.

Moving a role to another branch in the hierarchy

One benefit to moving a whole role is that any portal accounts simply move along with their parent role, and Salesforce doesn't have to change the related sharing. On the other hand, Salesforce must do all of the remaining work associated with Wendy's move for *every* user in the role being moved and for all their data.

Changing the owner of a portal account

The effort required for this task can be surprising because it looks like a simple data update—changing the name of the user in the Account owner field. When owners are in different roles, as the previous example shows, Salesforce isn't moving only the portal roles to a new parent role, however; it's also reparenting the portal roles associated with the account and adjusting sharing for all of the data associated with the portal account.

Summary

The simple examples and scenarios in this document provide a glimpse into the complex data processing and calculations that occur under the hood in Salesforce to support its robust record access mechanisms. As you can imagine, that complexity increases exponentially in large organizations, many of which have more than 10,000,000 account records, 7,000 users, 2,000 roles, and 1,000 territories. When designing a record access model in such organizations, keeping these fundamentals in mind can help you architect the most efficient Salesforce implementation for your organization's business needs.

For additional information on best practices for scaling Salesforce record access in your own organization, see *Designing Record Access for Enterprise Scale*.

Note: Email ccefeedback@salesforce.com to provide feedback on this document.